

TierMem: Balancing Compressed Memory and Raw Evidence for Long-Horizon Agent Memory

Qiming Zhu* Shunian Chen Rui Yu Zhehao Wu Benyou Wang
School of Data Science, The Chinese University of Hong Kong, Shenzhen
qimingzhu@link.cuhk.edu.cn

Abstract

Long-horizon agents must preserve past experience before knowing what future queries will require. Compact memory makes growing histories efficient to reuse by distilling experience into reusable abstractions, while raw history preserves fine-grained evidence for faithful grounding. Because future queries require different levels of detail, the appropriate memory granularity cannot be reliably fixed when memory is written. We therefore argue that memory granularity should be determined when the future information need becomes known. We introduce **TierMem**, a provenance-linked two-tier memory framework that allocates memory granularity according to evidence sufficiency. TierMem retrieves compact memory by default, uses a learned router to identify when finer-grained evidence is required, and follows provenance links to recover relevant raw evidence. Recovered evidence can be consolidated into compact memory for future reuse. Across two long-horizon memory benchmarks, TierMem recovers much of the quality enabled by raw evidence while substantially reducing inference cost. Further analyses show that provenance improves evidence recovery and that consolidation progressively increases the share of queries served by compact memory.

1 Introduction

Long-horizon agents accumulate experience before knowing how that experience will be used in the future (Chen et al., 2023; Liu et al., 2023). Compact memory distills growing histories into reusable abstractions and enables efficient access to past experience (Chhikara et al., 2025; Fang et al., 2025). Raw history preserves the source evidence behind those abstractions. Future queries can require different levels of detail. A memory representation chosen when experience is stored must therefore anticipate an information need that has not yet appeared.

This uncertainty creates a memory granularity problem. Compression determines which information remains directly accessible to future queries. A compact memory can retrieve the relevant past experience while lacking sufficient evidence for a faithful answer. Raw history preserves the missing evidence, but accessing raw history for every query gives up much of the efficiency gained through compression. The appropriate granularity therefore depends on the query rather than on the stored experience alone.

The arrival of a query reveals the information need that was unavailable during memory construction. Evidence sufficiency can then determine whether the compact representation already supports the query or whether more detailed evidence is required. This observation suggests a general principle for long-horizon memory. The system should preserve efficient abstractions together with recoverable source evidence and determine the required memory granularity when the query becomes known.

Existing memory systems improve how past experience is compressed, organized, and retrieved (Chhikara et al., 2025; Fang et al., 2025; Li et al., 2025). Adaptive retrieval systems

*Email: qimingzhu@link.cuhk.edu.cn

further control when additional retrieval or computation is useful (Asai et al., 2023; Du et al., 2025; Yan et al., 2025). Within this broader space, TierMem focuses on the sufficiency boundary between compact memory and its supporting evidence. It makes memory granularity an explicit inference-time decision and uses evidence sufficiency to control that decision.

We introduce **TierMem**, a provenance-linked two-tier memory framework for sufficiency-guided memory access. Tier-1 stores compact memory for efficient retrieval, and Tier-2 preserves the corresponding raw evidence. Provenance links connect compact memories to their source pages and provide a direct path for evidence recovery. A learned router evaluates whether retrieved Tier-1 evidence is sufficient for the current query. Queries with sufficient evidence remain on the compact path. Queries requiring more detail access the linked raw evidence and can perform bounded additional retrieval. Verified evidence can subsequently be consolidated into Tier-1 while preserving provenance, allowing information that proves useful in practice to become cheaper to access in the future.

Across **LoCoMo** and **LongMemEval**, TierMem recovers much of the answer quality available from raw-grounded memory while substantially reducing inference cost. Relative to raw grounding, TierMem reduces average input tokens by **34–49%** and latency by **34–66%** across the two benchmarks. Further analyses show that provenance improves evidence recovery and that consolidation progressively increases the share of queries served by compact memory. Together, these results support query-time granularity allocation as an effective way to combine efficient memory reuse with access to detailed evidence.

Contributions.

1. We formulate long-horizon agent memory as a **query-time granularity allocation problem**. Future information needs are unknown when memory is constructed, while evidence sufficiency becomes observable when the query arrives.
2. We introduce **TierMem**, a provenance-linked two-tier memory framework that connects compact memory with recoverable raw evidence through learned sufficiency routing and evidence-backed consolidation.
3. We show that adaptive memory granularity preserves strong answer quality while substantially reducing the inference cost of raw-grounded access, and that provenance and consolidation improve the effectiveness of this allocation over repeated use.

2 The memory granularity problem

Future queries can require different levels of detail from the same past experience. One query may depend on a high-level event, preference, or prior decision. Another may depend on an exact name, number, timestamp, quotation, or exception from the same interaction. A long-horizon memory system must support these different information needs over a shared history.

Memory construction occurs before these future requirements are known. Compact memory makes repeated access efficient by retaining salient information in a reusable form. The construction process also determines which distinctions remain directly accessible. A detail omitted during memory construction cannot be recovered by retrieving the compact representation more accurately. Future use therefore introduces a granularity mismatch whenever a query requires information finer than the stored representation preserves.

We measure this mismatch with a controlled diagnostic on LoCoMo. We identify 327 atomic information items that remain recoverable from the raw conversation and test whether the same items remain recoverable from the corresponding Tier-1 memory. Tier-1 preserves 232 items, leaving a **29.1% future-query miss rate**. The misses vary across information types. Exact temporal information has the highest miss rate at **44.0%**, while names and numbers have miss rates of **28.7%** and **27.0%**. The result shows that compact memory can preserve the relevant experience while losing distinctions required by a later query.

Preserving raw history keeps finer evidence available after compact memory has been constructed. The value of accessing raw evidence depends on the query. Many queries

Information type	Raw	Compressed	Miss rate
Names	148	105	28.7%
Numbers	100	73	27.0%
Quotes	29	26	10.3%
Times	50	28	44.0%
Overall	327	232	29.1%

Table 1: **Information retained across memory granularities.** All 327 information items remain recoverable from raw history, while 232 remain recoverable from compressed memory. Exact details, especially temporal information, are more likely to be omitted.

already have sufficient support in compact memory. Queries that depend on an omitted detail require a finer view of the same experience. Uniform raw access would spend additional inference budget on queries whose evidence needs are already satisfied. The useful memory granularity is therefore query dependent.

The query provides the information that was unavailable during memory construction. Once the query arrives, the system can evaluate whether the retrieved compact memory contains sufficient evidence for the required answer. Sufficient evidence supports direct use of compact memory. Insufficient evidence creates a reason to recover finer source information. Evidence sufficiency therefore provides a practical signal for allocating memory granularity at inference time.

A memory architecture built around this signal requires three capabilities. First, it should preserve compact memory together with traceable access to finer source evidence. Second, it should use query-specific evidence sufficiency to determine which granularity to access. Third, evidence recovered at finer granularity should improve future compact access while preserving its source. TierMem realizes these capabilities through provenance-linked memory tiers, learned sufficiency routing, and evidence-backed consolidation.

3 TierMem: Adaptive Memory Granularity Allocation

TierMem implements adaptive memory granularity through three coupled mechanisms. It preserves past experience at two linked granularities, uses evidence sufficiency to select the granularity required by each query, and consolidates recovered evidence for future reuse. Figure 1 gives an overview.

3.1 Provenance-Linked memory tiers

TierMem preserves past experience at two linked granularities. Tier-1 provides compact memory for efficient reuse, while Tier-2 preserves the source information needed when a future query requires finer detail. Provenance links connect the two representations so that compact retrieval also identifies where supporting evidence can be recovered.

Tier-2 raw memory. Incoming interaction turns are accumulated into fixed-size pages. When a page P_i is sealed, TierMem stores the original interaction content in Tier-2 memory $\mathcal{M}_R$. Tier-2 pages remain immutable within the evaluation horizon. Stable page identifiers allow later memory entries and recovered evidence to refer to the same source interaction throughout the memory lifecycle.

Tier-1 compact memory. Each sealed page is summarized into compact memory entries and stored in Tier-1 memory $\mathcal{M}_S$. A Tier-1 entry contains its memory text, a dense embedding for retrieval, and provenance links ρ_i to the Tier-2 pages that support it. Tier-1 therefore provides an efficient retrieval representation while retaining a direct path to finer source evidence.

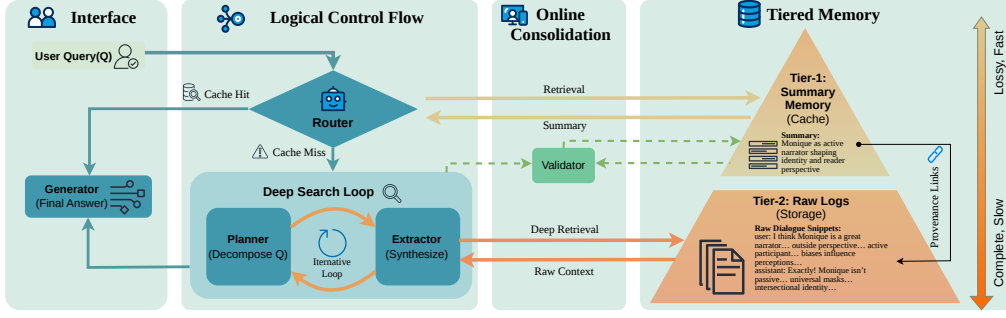


Figure 1: **TierMem overview.** TierMem stores compact memory in Tier-1 and source evidence in Tier-2. Each Tier-1 entry retains provenance links ρ to its supporting raw pages. Given a query, TierMem first retrieves Tier-1 memory and applies a learned sufficiency router. Sufficient Tier-1 evidence supports direct answering. Insufficient evidence triggers provenance-guided access to Tier-2 and bounded additional retrieval when needed. Recovered evidence can be consolidated into Tier-1 while preserving provenance.

The write path constructs both granularities together. For each sealed page P_i , TierMem stores the raw page in $\mathcal{M}_R$, generates the corresponding compact entries, indexes them in $\mathcal{M}_S$, and records the page identifier in their provenance metadata. The resulting memory state preserves reusable abstractions without severing their connection to the interactions from which they were derived.

3.2 Sufficiency-Guided memory access

Given a query, TierMem uses evidence sufficiency to determine which memory granularity to access. The system begins with compact memory because many queries can be answered from Tier-1 alone. Raw evidence is accessed when the retrieved compact memory does not provide enough information for the current query.

Summary sufficiency routing. For a query Q , TierMem first retrieves the top- k Tier-1 entries

$$Z_0 = \text{Retrieve}(\mathcal{M}_S, Q).$$

A lightweight router then predicts whether Z_0 contains sufficient evidence:

$$a \sim \pi_\theta(a \mid Q, Z_0), \quad a \in S, R.$$

The S action keeps the query on the compact path and generates the answer from Z_0 . The R action requests finer evidence from Tier-2.

The routing criterion is evidence sufficiency. Retrieval identifies memories related to the query, while the router determines whether those memories contain the information required for answering. A retrieved memory may identify the correct event or topic while omitting a required entity, timestamp, number, or other detail. TierMem escalates when such information is needed.

Learning the sufficiency router. We train the router from paired executions of the compact and raw-grounded paths. For each training query, TierMem first evaluates whether the retrieved Tier-1 evidence explicitly supports the answer. Queries with sufficient Tier-1 evidence receive an S label. When Tier-1 is insufficient and raw-grounded execution successfully recovers the answer, the query receives an R label. Cases where raw-grounded execution also fails are excluded because they do not provide supervision about the value of escalation.

Training proceeds in two stages. Supervised fine-tuning first teaches the router the sufficiency decision and output protocol. We then apply GRPO to refine the routing policy under an accuracy–cost objective. The reward favors correct routing decisions and penalizes unnecessary use of the raw path. Full data construction, reward specification, prompts, and hyperparameters are provided in Appendix D.

Provenance-guided escalation. When the router selects R, TierMem follows the provenance links attached to the retrieved Tier-1 entries and reopens their supporting Tier-2 pages. The compact retrieval result therefore serves two roles. It identifies semantically relevant past experience and provides direct access to the finer evidence behind that experience.

The recovered pages are integrated into an evidence set for the current query. Provenance-guided access concentrates the initial raw retrieval on pages already connected to relevant compact memories.

Bounded evidence expansion. Linked pages may still leave part of the information need unresolved. TierMem therefore supports bounded additional retrieval over compact memory and raw pages. The system alternates evidence integration with a planning step that decides whether the available evidence is sufficient or further search is needed. The loop terminates when the evidence is sufficient or when it reaches the maximum number of retrieval rounds. We use $T_{\max} = 3$ in the main experiments. The final answer is generated from the assembled evidence set.

3.3 Evidence-Backed memory consolidation

Evidence recovered through escalation provides a signal for improving future compact memory. A detail that was absent from Tier-1 and later proved useful for answering a query has demonstrated value beyond the original memory construction stage. TierMem uses such evidence to update the compact tier while retaining its connection to the source history.

After an escalated query, TierMem extracts grounded facts from the recovered evidence and associates each fact with its supporting Tier-2 pages. For each candidate fact, the system retrieves related Tier-1 entries and selects one of four memory operations: ADD, UPDATE, SKIP, or DELETE. ADD creates a new compact memory when the recovered information is distinct. UPDATE enriches an existing entry with newly recovered detail. SKIP leaves Tier-1 unchanged when the information is already represented. DELETE removes redundant or outdated Tier-1 entries.

Updates preserve provenance. An updated Tier-1 entry inherits links from both the existing memory and the newly recovered evidence. Tier-2 pages remain unchanged. The compact tier can therefore evolve while its contents remain traceable to the underlying interaction history.

Consolidation allows future memory construction to benefit from observed information needs. Details that repeatedly matter to later queries can become directly accessible in Tier-1, reducing the need to recover the same evidence again. TierMem therefore couples query-time granularity allocation with a memory update process that progressively adapts the compact representation to demonstrated future use.

4 Experiments

We evaluate TierMem on two long-horizon memory benchmarks and compare it against both summary-centric baselines and raw/controller-centric systems. Our goal is to test a single question: does inference-time evidence allocation improve the accuracy–efficiency frontier relative to static memory-access policies?

Dataset	System	Acc. $\uparrow$	UOR $\downarrow$	Tok _{in} $\downarrow$	Lat. $\downarrow$
LoCoMo	Mem0	0.684	23.3	1085.9	20.32
	LightMem	0.773	14.7	1159.5	27.69
	O-mem	0.764	16.3	2178.1	1.16
	MemOS	0.753	17.1	2692.3	10.80
	MemR3	0.860	–	4585.7	12.30
	GAM	0.869	–	15773.7	21.85
	TierMem (summary-only)	0.755	15.4	260.1	1.24
	TierMem (raw-grounded)	0.873	–	7398.4	17.18
	TierMem (router)	0.854	–	3742.1	5.81
LongMemEval	Mem0	0.596	29.6	1268	15.2
	LightMem	0.716	16.8	1160	7.87
	O-mem	0.620	25.7	2680	12.3
	MemOS	0.652	27.4	1682	2.3
	MemR3	0.742	–	6347	18.9
	GAM	0.764	–	37850	42.3
	TierMem (summary-only)	0.678	19.6	380	5.37
	TierMem (raw-grounded)	0.808	–	7298	13.56
	TierMem (router)	0.752	–	4803	9.01

Table 2: **Main results on LoCoMo and LongMemEval.** The main paper keeps only the metrics needed to show the operating point of each system; full subtype breakdowns are reported in the appendix.

4.1 Benchmarks, metrics, and compared systems

We evaluate on **LoCoMo** and **LongMemEval**, two benchmarks that stress long-horizon conversational memory under question types such as recall, temporal reasoning, multi-session aggregation, and knowledge updates (Maharana et al., 2024; Wu et al., 2025). Together, they provide a useful test bed for our main claim: long-horizon memory systems must support both efficient coarse recall and query-specific recovery of fine-grained evidence.

Our main tables report four metrics: **accuracy**, **UOR**, **average input tokens per query**, and **average end-to-end latency**. UOR (**Unverifiable Omission Rate**) measures failures caused by missing answer-supporting evidence in compressed memory, which can be found in TierMem(raw-grounded). In router-specific analyses, we additionally report **R-rate**, **Acc.@R**, and **Recall@hard**.

We compare against representative summary-centric systems (**Mem0**, **LightMem**, **O-mem**, **MemOS**) and raw/controller-centric systems (**MemR3**, **GAM**) (Chhikara et al., 2025; Fang et al., 2025; Wang et al., 2025; Li et al., 2025; Du et al., 2025; Yan et al., 2025). For TierMem, we report three main operating points: **summary-only**, **raw-grounded**, and **router**. All TierMem variants share the same generator, embedding model, reranker, and retrieval stack; they differ only in whether Tier-2 is consulted and how escalation is triggered. Additional implementation details, stronger routing references, and full subtype results are reported in the appendix.

4.2 Main results

Table 2 shows that TierMem reaches a favorable operating point on both benchmarks. On **LoCoMo**, the routed variant lands close to the always-raw accuracy reference while preserving a large fraction of the efficiency benefit of summary-first access. Relative to **TierMem (summary-only)**, routing improves accuracy by **9.9** points. Relative to **TierMem (raw-grounded)**, it preserves **97.8%** of raw-grounded accuracy while reducing average input tokens by **49.4%** and latency by **66.2%**.

On **LongMemEval**, which is more demanding for compact memory because it stresses updates, temporal reasoning, and multi-session aggregation, TierMem (router) raises accuracy from **0.678** to **0.752** over the summary-only variant. At the same time, it remains

Method	Acc.	R-rate	Acc.@R	Tok _{in}	Lat.
Raw-grounded reference	0.873	100%	87.3	7398	17.18
No-Linked	0.836	38.0%	77.5	3832	6.33
Linked (ours)	0.854	38.1%	82.5	3742	5.81

Table 3: **Effect of provenance links.** Improvements are concentrated on escalated queries, not on the summary-only path.

System	R-rate	Recall@hard	Acc.	Router tok.	Router Lat.(s)
TierMem (summary-only)	0%	–	0.755	–	–
TierMem (raw-grounded)	100%	100.0%	0.873	–	–
Zero-shot router	20.6%	44.7%	0.806	901	1.025
SFT router	38.2%	65.8%	0.831	680	0.421
SFT+GRPO router	38.1%	76.1%	0.854	678	0.376

Table 4: **Router behavior on LoCoMo.** Full training details and additional counterfactual analyses are reported in the appendix.

substantially cheaper than the always-raw reference, reducing average input tokens by **34.2%** and latency by **33.6%**.

Taken together, the main results support a single systems conclusion: selective escalation recovers much of the fidelity lost under summary-only access without paying the full cost of always reopening raw history.

4.3 Mechanism analyses

We next isolate two components that determine where TierMem sits on the accuracy–efficiency frontier: provenance-guided escalation and sufficiency routing.

4.3.1 Provenance links improve recovery after escalation

A key design choice in TierMem is that summary units point back to supporting raw pages. To isolate this mechanism, we compare **Linked** TierMem against a **No-Linked** variant that still escalates but does not prioritize linked raw pages during escalation.

Linked TierMem improves end-to-end accuracy by **1.8** points over No-Linked, with nearly identical routing rates. The gain is concentrated on escalated queries: **Acc.@R** rises from **77.5%** to **82.5%**. This is the expected role of provenance links. They do not change what summaries contain; they improve how the system finds the right raw evidence once escalation happens.

Among the **378** LoCoMo queries that the summary-only system answers incorrectly, **45** are corrected after the system consults the *linked* raw pages. This supports the claim that provenance links are not merely bookkeeping: they measurably improve evidence recovery on otherwise failed queries.

4.3.2 The router is conservative and effective

Table 4 shows that training improves the router’s decision boundary, especially by increasing recall on hard queries while keeping controller overhead modest.

Training shifts the router to a much better operating point. Relative to zero-shot control, SFT+GRPO improves **Recall@hard** from **44.7%** to **76.1%** and end-to-end accuracy from **0.806** to **0.854**, while also reducing router latency. The learned controller is therefore not only more accurate, but also more efficient than a naive prompting baseline.

The router is also conservative in a useful direction. Its dominant error mode is unnecessary escalation rather than missed escalation, which is usually the safer systems failure

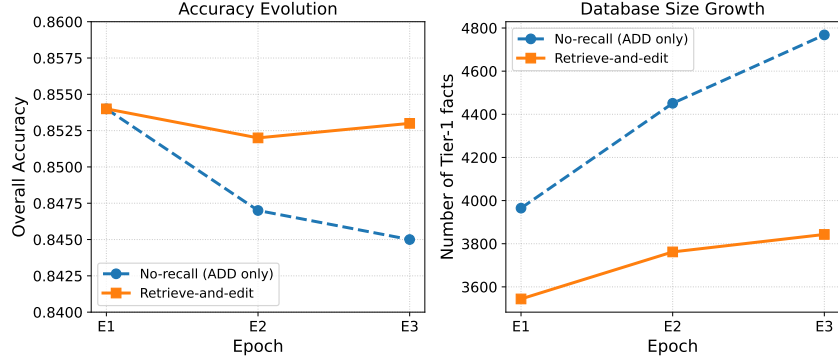


Figure 2: Evolution of TierMem under replayed write-back on LoCoMo. Summary-path traffic increases over epochs while accuracy remains stable and the governed retrieve-and-edit variant slows Tier-1 growth relative to append-only write-back.

mode because it trades efficiency for verification rather than replacing raw evidence with unsupported guesses. The controller overhead also remains modest relative to full raw grounding.

5 Analysis

Section 4 shows that TierMem improves the accuracy–efficiency frontier, and the mechanism analyses isolate why provenance links and routing help. We now turn to two systems questions that only arise once the architecture is used repeatedly: when the extra write path pays off, and whether write-back actually converts past expensive verification into future cheap access.

5.1 Amortized cost over repeated access

TierMem incurs a shared write cost to construct provenance-linked compact and raw memory, while its query-time savings accumulate over subsequent accesses. This makes amortized cost a natural measure of the system’s long-term efficiency. Tier-1 provides the efficient access path, while Tier-2 preserves recoverable source evidence for queries that require finer detail.

On LoCoMo, the shared TierMem write phase consumes **1226.61k** input tokens. Routed access saves approximately **3656.3** input tokens per query relative to always-raw grounding. Across the **1540** evaluated queries, these savings total approximately **5.63M** input tokens, more than **4.5×** the input-token cost of memory construction. The result shows that the upfront cost of maintaining linked memory can be amortized through repeated query-time access.

5.2 Write-back Turns Past Escalations into Future Cheap Access

We evaluate write-back in an epoch-wise replay protocol. Within each epoch, Tier-1 is frozen while the system answers the full query set. Verified findings from escalations are logged during the epoch and written back only between epochs. This keeps evaluation conditions within an epoch fixed while revealing whether write-back reduces future reliance on the raw path.

The replay study shows the intended qualitative effect: summary-path traffic increases over replay while overall accuracy remains stable. Under *retrieve-and-edit* write-back, summary-path traffic rises from **932** queries in E1U to **1224** in E3U, a **31.3%** increase, while overall accuracy remains essentially unchanged (**0.854** $\rightarrow$ **0.853**). Over the same replay, average

input tokens fall from **3958** to **2447** (**38.2%** reduction) and latency falls from **5.14s** to **3.85s** (**25.1%** reduction).

Crucially, governance matters. Compared with the append-only *no-recall write-back* setting, retrieve-and-edit reaches a comparable final accuracy but with a much smaller Tier-1 footprint: at epoch 3, retrieve-and-edit stores **3843** Tier-1 facts versus **4768** for append-only write-back, a reduction of **925** facts (**19.4%**). In other words, write-back helps most when it behaves like consolidation rather than naive cache accretion.

6 Related Work

6.1 Write-time compressed memory

Long-horizon LLM agents often rely on external memory because feeding full histories into the context window is costly and can be brittle [Chen et al. \(2023\)](#); [Liu et al. \(2023\)](#). Many memory systems therefore compress interaction histories into summaries, facts, graphs, or other compact units [Packer et al. \(2024\)](#); [Zhong et al. \(2023\)](#); [Chhikara et al. \(2025\)](#); [Fang et al. \(2025\)](#); [Rasmussen et al. \(2025\)](#). These systems improve efficiency, but they generally treat memory quality as a write-time compression problem. TierMem instead focuses on how the appropriate memory granularity can be determined when the downstream information need becomes known.

6.2 Adaptive retrieval and control

Recent work in adaptive retrieval shows that retrieval and additional computation can be conditioned on the information need rather than invoked uniformly [Jiang et al. \(2023\)](#); [Asai et al. \(2023\)](#); [Shinn et al. \(2023\)](#). Closer to agent memory, systems such as MemR³ and GAM allocate additional inference-time budget to gather stronger evidence under cost constraints [Du et al. \(2025\)](#); [Yan et al. \(2025\)](#). TierMem applies this adaptive-access principle to selecting between compact memory and provenance-linked raw evidence according to evidence sufficiency.

6.3 Provenance, grounding, and memory consolidation

Prior retrieval systems have explored source-linked representations, citation, revision, and verifiable grounding, including graph-based RAG systems such as GraphRAG [Edge et al. \(2025\)](#); [Gao et al. \(2022\)](#); [Cao & Wang \(2024\)](#). Agent learning systems also update reusable knowledge from accumulated experience; for example, ExpeL iteratively maintains natural-language insights extracted from task trajectories [Zhao et al. \(2024\)](#). TierMem brings these ideas together in a two-tier long-horizon memory architecture, where compact memories remain linked to raw evidence, evidence sufficiency controls when finer-grained evidence is accessed, and recovered evidence can be consolidated back into compact memory while retaining provenance.

7 Discussion

The memory granularity problem also arises when agents reuse past experience to guide future actions. In coding and other evolving workflows, compact memory can preserve reusable strategies, recurring failure modes, and prior decisions, while raw trajectories preserve exact tests, tool outputs, edits, and environment-specific evidence. Different tasks can require different levels of detail from the same experience.

TierMem can operate as a long-term memory access layer in such systems. The downstream agent retains its own planning, tool use, and working memory, while TierMem determines whether retrieved long-term experience should be consumed in compact form or reopened at finer granularity. Evidence recovered during use can then be consolidated for cheaper future access. The same sufficiency-guided principle can extend beyond two tiers to richer memory hierarchies with multiple levels of abstraction.

8 Conclusion

TierMem formulates long-horizon agent memory as a query-time granularity allocation problem. It preserves compact memory together with provenance-linked raw evidence, uses learned evidence sufficiency to select the required granularity, and consolidates recovered evidence for future reuse. Across LoCoMo and LongMemEval, TierMem preserves strong answer quality while substantially reducing the inference cost of raw-grounded access. These results show that memory granularity can be allocated according to the information need revealed at use time.

References

- Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection, 2023. URL <https://arxiv.org/abs/2310.11511>.
- Shuyang Cao and Lu Wang. Verifiable generation with subsentence-level fine-grained citations, 2024. URL <https://arxiv.org/abs/2406.06125>.
- Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation, 2023. URL <https://arxiv.org/abs/2306.15595>.
- Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory, 2025. URL <https://arxiv.org/abs/2504.19413>.
- Xingbo Du, Loka Li, Duzhen Zhang, and Le Song. MemR³: Memory retrieval via reflective reasoning for LLM agents, 2025. URL <https://arxiv.org/abs/2512.20237>.
- Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitanaky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization, 2025. URL <https://arxiv.org/abs/2404.16130>.
- Jizhan Fang, Xinle Deng, Haoming Xu, Ziyang Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, Huajun Chen, and Ningyu Zhang. LightMem: Lightweight and efficient memory-augmented generation, 2025. URL <https://arxiv.org/abs/2510.18866>.
- Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan, Vincent Y. Zhao, Ni Lao, Hongrae Lee, Da-Cheng Juan, and Kelvin Guu. RARR: Researching and revising what language models say, using language models, 2022. URL <https://arxiv.org/abs/2210.08726>.
- Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in LLM agents via incremental multi-turn interactions, 2025. URL <https://arxiv.org/abs/2507.05257>.
- Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation, 2023. URL <https://arxiv.org/abs/2305.06983>.
- Zhiyu Li, Chenyang Xi, Chunyu Li, Ding Chen, Boyu Chen, Shichao Song, Simin Niu, Hanyu Wang, Jiawei Yang, Chen Tang, Qingchen Yu, Jihao Zhao, Yezhaohui Wang, Peng Liu, Zehao Lin, Pengyuan Wang, Jiahao Huo, Tianyi Chen, Kai Chen, Kehang Li, Zhen Tao, Huayi Lai, Hao Wu, Bo Tang, Zhengren Wang, Zhaoxin Fan, Ningyu Zhang, Linfeng Zhang, Junchi Yan, Mingchuan Yang, Tong Xu, Wei Xu, Huajun Chen, Haofen Wang, Hongkang Yang, Wentao Zhang, Zhi-Qin John Xu, Siheng Chen, and Feiyu Xiong. MemOS: A memory OS for AI system, 2025. URL <https://arxiv.org/abs/2507.03724>.
- Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts, 2023. URL <https://arxiv.org/abs/2307.03172>.

- Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents, 2024. URL <https://arxiv.org/abs/2402.17753>.
- Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems, 2024. URL <https://arxiv.org/abs/2310.08560>.
- Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. Zep: A temporal knowledge graph architecture for agent memory, 2025. URL <https://arxiv.org/abs/2501.13956>.
- Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL <https://arxiv.org/abs/2303.11366>.
- Piaohong Wang, Motong Tian, Jiaxian Li, Yuan Liang, Yuqing Wang, Qianben Chen, Tiannan Wang, Zhicong Lu, Jiawei Ma, Yuchen Eleanor Jiang, and Wangchunshu Zhou. O-mem: Omni memory system for personalized, long horizon, self-evolving agents, 2025. URL <https://arxiv.org/abs/2511.13593>.
- Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Long-MemEval: Benchmarking chat assistants on long-term interactive memory, 2025. URL <https://arxiv.org/abs/2410.10813>.
- B. Y. Yan, Chaofan Li, Hongjin Qian, Shuqi Lu, and Zheng Liu. General agentic memory via deep research, 2025. URL <https://arxiv.org/abs/2511.18423>.
- Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners, 2024. URL <https://arxiv.org/abs/2308.10144>.
- Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory, 2023. URL <https://arxiv.org/abs/2305.10250>.

A Additional Experimental Details and Full Tables

This appendix collects material that should not dominate the main paper: full subtype results, output-token statistics, detailed latency notes, routing-cost breakdowns, and additional write-back and provenance analyses.

A.1 Full LoCoMo Results

System	Accuracy (LLM-Judge) $\uparrow$					UOR % $\downarrow$	F1 $\uparrow$	Efficiency		
	Multi	Temp.	Open	Single	Overall	Rate	Overall	Tok $_{in}$ $\downarrow$	Tok $_{out}$ $\downarrow$	Lat.(s) $\downarrow$
Mem0 Chhikara et al. (2025)	0.628	0.707	0.531	0.712	0.684	23.3	0.426	1085.9	8.0	20.32
LightMem Fang et al. (2025)	0.709	0.807	0.542	0.810	0.773	14.7	0.495	1159.5	6.9	27.69
O-mem Wang et al. (2025)	0.706	0.776	0.604	0.797	0.764	16.3	0.512	2178.1	544.2	1.16
MemOS Li et al. (2025)	0.773	0.639	0.625	0.804	0.753	17.1	0.400	2692.3	13.3	10.80
MemR3 Du et al. (2025)	0.812	0.829	0.729	0.902	0.860	–	0.586	4585.7	384.1	12.30
GAM Yan et al. (2025)	0.858	0.860	0.781	0.886	0.869	–	0.574	15773.7	1231.5	21.85
TierMem (summary-only)	0.656	0.717	0.667	0.812	0.755	15.4	0.514	260.1	6.1	1.24
TierMem (raw-grounded)	0.812	0.863	0.719	0.916	0.873	–	0.604	7398.4	521.3	17.18
TierMem (router)	0.798	0.838	0.708	0.895	0.854	–	0.589	3742.1	335.4	5.81
TierMem (router, GPT-4.1-mini)	0.755	0.850	0.677	0.904	0.851	–	0.591	3620.9	235.4	6.40

Table 5: Full LoCoMo results with subtype breakdown, output tokens, and latency.

A.2 Full LongMemEval Results

Method	SS-User	SS-Asst	SS-Pref	Multi-S	Know. Upd	Temp. Reas	Overall	UOR % $\downarrow$	Tok _{in} /Q	Tok _{out} /Q	Lat./Q
Mem0	0.800	0.589	0.533	0.571	0.667	0.489	0.596	29.6	1268	9	15.2
LightMem	0.871	0.339	0.767	0.790	0.885	0.700	0.716	16.8	1160	7	7.87
O-mem	0.957	0.446	0.773	0.632	0.551	0.542	0.620	25.7	2680	670	12.3
MemOS	0.872	0.617	0.963	0.514	0.840	0.557	0.652	27.4	1682	206	2.3
MemR3	0.929	0.929	0.600	0.684	0.769	0.639	0.742	–	6347	437	18.9
GAM	0.957	0.964	0.600	0.752	0.756	0.629	0.764	–	37850	1448	42.3
TierMem (summary-only)	0.900	0.857	0.833	0.549	0.756	0.519	0.678	19.6	380	11	5.37
TierMem (raw-grounded)	0.971	0.982	0.900	0.752	0.846	0.662	0.808	–	7298	677	13.56
TierMem (router)	0.957	0.964	0.833	0.707	0.769	0.602	0.752	–	4803	439	9.01
TierMem (router, GPT-4.1-mini)	0.929	0.982	0.867	0.714	0.782	0.624	0.770	–	4574	328	9.74

Table 6: Full LongMemEval results with subtype breakdowns.

A.3 Write-phase cost on LoCoMo

System	Write Tok _{in} (k)	Write Tok _{out} (k)
Mem0	1642.21	249.13
LightMem	79.24	20.53
O-Mem	1411.87	176.26
MemOS	472.51	130.28
GAM	1063.64	103.13
TierMem	1226.61	132.26

Table 7: Write-phase cost on LoCoMo.

A.4 Provenance Linking Statistics

TierMem maintains bidirectional provenance links between Tier-1 memories and Tier-2 raw pages. During the QA phase, retrieved memories can therefore be traced back to their source pages for verification or escalation.

Across the 794 queries for which the current logging stack records provenance details, TierMem linked **2,724** facts to downstream answers, or **3.43** facts per logged query on

Metric	Value
Total sessions	10
Total conversation turns	2,674
Total pages (Tier-2 chunks)	265
Total memories (Tier-1 facts)	3,245
Tier-2 / Tier-1 serialized storage ratio	$1.76\times$
Tier-1 as a share of Tier-2 storage	56.9%

Table 8: **TierMem storage analysis (LoCoMo sessions).**

Component	Model / Backbone	Role	Notes
Answer generator	GPT-4.1-mini	Final answer generation	Shared by all TierMem variants in the main comparison
Router (ours)	Qwen3-0.6B	Summary sufficiency control	LoRA SFT + GRPO
Embedding model	text-embedding-3-small	Tier-1 retrieval	Shared across all TierMem variants
Reranker	Qwen3-Reranker-0.6B	Optional reranking	Shared across all TierMem variants

Table 9: **System transparency for TierMem.** The goal of this table is to make the claimed “lightweight” routing overhead auditable and reproducible.

average. Each linked fact carries pointers to the corresponding Tier-2 pages, so the system can selectively reopen authoritative raw evidence without replaying the full interaction history. These links are also preserved during write-back, which keeps consolidated Tier-1 units auditable rather than turning them into untraceable cache state.

A.5 Memory footprint and storage efficiency

TierMem maintains two storage tiers: Tier-1 stores compressed summaries and extracted facts, while Tier-2 preserves immutable raw conversation pages. Table A.5 summarizes the resulting footprint.

The $1.76\times$ compression ratio is moderate rather than aggressive. This reflects a deliberate design choice: Tier-1 stores semantically rich memory units—facts, events, and entity updates with provenance—rather than highly abstractive summaries optimized only for compression. As a result, Tier-1 alone uses 56.9% of the storage of the raw tier, yet still supports a strong cheap path: summary-only access answers 75.5% of LoCoMo queries correctly. This storage ratio is close to, but distinct from, the $1.81\times$ raw-to-summary text compression measured in the controlled future-query miss diagnostic, because the two analyses use different denominators and measure different stages of the pipeline.

This storage profile also clarifies the systems trade-off. TierMem is not trying to minimize bytes at all costs; it keeps the raw tier intact for verifiability and adds a compact but information-rich summary tier to reduce query-time cost. The benefit shows up at inference time: across queries with provenance logs, each answer links to an average of 3.43 Tier-1 facts, which can be traced back to specific Tier-2 pages when escalation is needed.

B Implementation Details

B.1 Memory Instantiation

Our framework is general; in experiments we instantiate the memory management system using the open-source **Mem0** library [Chhikara et al. \(2025\)](#) for vector indexing and storage. We adapt Mem0 to our **Paged-Log Architecture** as follows:

- **Summarization prompt (f_{summ}).** We replace Mem0’s default entity-graph extraction with **page-level summarization**. The prompt instructs the model to generate a concise, self-contained summary of the interaction log, preserving key dates, decisions, and user preferences.
- **Storage schema.** Each page summary is embedded and indexed in the vector store. We extend Mem0 metadata to include the Page ID (P_i) as a provenance pointer ρ_i , enabling retrieval to map back from a summary vector to its originating page.
- **Embedding model.** We embed summaries with text-embedding-3-small.

Mem0 components used vs. disabled. In our implementation, Mem0 is used only for (i) storing and querying embeddings for Tier-1 summary entries and (ii) attaching/retrieving metadata (e.g., `raw_log_id/page_id`) for provenance. We do not use Mem0’s entity-graph construction, graph updates, or graph-based retrieval; these components are disabled/bypassed so that Tier-1 consists solely of page summaries plus metadata pointers.

B.2 Paged-Log Ingestion and Persistence

We implement a **paged ingestion** procedure that converts streaming dialogue turns into fixed-size *pages* stored in a persistent PageStore. Each incoming turn is appended to the current page with an explicit timestamp and speaker tag. Once a page reaches a configured maximum length (default: 1000 tokens), we trigger a summarization-and-indexing step via `mem0.add(infer=True)` over the entire page.

Page format. Each page stores raw interaction text line-by-line in the canonical form:

[timestamp] speaker : text.

This preserves multi-speaker context for datasets with more than one interlocutor.

Provenance. For each page P_i , we generate a unique `page_id`. We set `raw_log_id` $\leftarrow$ `page_id` in Mem0 metadata such that any retrieved summary hit can be deterministically mapped back to the original raw page content in PageStore.

Durability and replay. PageStore persists pages to disk and supports session-scoped loading. This enables resuming runs without re-ingesting all prior turns and ensures that the raw evidence used for research-mode grounding remains accessible.

B.3 Tier-2 compression and storage policy

TierMem distinguishes *authoritative evidence* from *retrieval acceleration*. A sealed Tier-2 page is immutable at the semantic level: the underlying raw text remains the source of truth for later verification. However, the access path to that page can still be optimized. In principle, Tier-2 can be paired with compressed vector indexes, quantized ANN structures, hot/warm/cold storage, or archival page stores, as long as the page content itself remains recoverable when a provenance link demands reopening it.

This distinction matters conceptually. TierMem does not require every raw page to stay in the most expensive online representation. It requires only that evidence compression happen at the index or storage layer rather than by rewriting or abstracting away the evidentiary content itself. In deployment, page-level archival, TTL, or cold-storage policies can be applied as storage-management operations, provided that page contents remain recoverable whenever a provenance link demands reopening them. In the current experiments, we keep Tier-2 fully replayable in order to isolate the memory-allocation question from systems-engineering optimizations.

B.4 Deferred Summarization for Unindexed Pages

To handle sessions that terminate before a page fills (or failures during ingestion), we provide a **deferred summarization** routine `auto_summary`. It enumerates pages whose summaries

have not yet been stored and calls `mem0.add(infer=True)` for each such page. The returned memory strings are recorded in `PageStore` and the page is marked as indexed, ensuring that the vector store remains consistent with the persistent log.

B.5 Query Answering Pipeline

Given a query q , we first retrieve top- k summary hits from `Mem0` using vector similarity search. Each hit contains a summary text and a provenance pointer to its source page.

Two-path routing (S/R). We use a lightweight router to select between:

- **S-path (Summary-only).** When retrieved summaries contain an explicit answer, we generate the response directly from the summaries.
- **R-path (Research).** When summaries are ambiguous or incomplete, we switch to a slower research mode that grounds generation in raw page evidence.

Cost accounting. We record (i) `Mem0` retrieval token usage and latency, (ii) router token usage, and (iii) answer-generation token usage. These are aggregated into online cost metrics for analysis.

B.6 Research Mode: Integration–Plan Loop

The R-path implements an iterative loop that alternates between **integration** (extracting grounded facts from evidence) and **planning** (deciding whether to retrieve more evidence).

Evidence construction. We group retrieved hits by `page_id` and build an evidence block that includes: (i) `Mem0` page summaries (when available) and (ii) the full raw page content from `PageStore`. This ensures that research mode always has access to verbatim text for grounding.

Integration step. The integration prompt extracts a set of *linked facts* of the form:

`fact, evidence_quote.`

Crucially, we *do not* ask the model to generate provenance identifiers.

Deterministic provenance linking. After integration, we map each extracted fact to its source pages using a deterministic post-processing step:

- First, we attempt substring matching of the normalized `evidence_quote` against each page’s raw content (and, if needed, its summaries).
- If quote matching fails, we fall back to keyword overlap between the fact text and page content.

This produces `source_pages` and `evidence_snippets` without relying on model-generated page identifiers, reducing hallucinated provenance.

Planning step. A separate planning prompt evaluates coverage and returns either `DONE` or `SEARCH` with retrieval commands. We support two retrieval operators:

- **MEM0_SEARCH:** semantic retrieval over `Mem0` summaries.
- **KEYWORD_SEARCH:** BM25-style keyword search over raw pages in `PageStore`.

We deduplicate issued commands and terminate early when no new pages are discovered.

B.7 Dynamic Write-back Protocol

We implement online consolidation as a closed-loop memory update step. Given a set of *verified* linked facts extracted during the Research Integration phase:

1. Conflict Retrieval. For each new candidate fact f_{new} , we query $\mathcal{M}_S$ to retrieve the top- k (e.g., $k = 3$) most similar existing summary units, denoted as $\mathcal{M}_{retrieved}$.

2. Operation Selection. We prompt an LLM (the *Memory Manager*) to compare f_{new} with $\mathcal{M}_{retrieved}$. The model outputs one of four actions:

- **SKIP:** If f_{new} is semantically entailed by $\mathcal{M}_{retrieved}$.
- **UPDATE:** If f_{new} adds specific details (e.g., timestamps, names) to a vague existing entry. The model generates the merged text.
- **ADD:** If f_{new} represents a distinct, previously unindexed event.
- **DELETE:** If $\mathcal{M}_{retrieved}$ is redundant or outdated. This operation removes or suppresses Tier-1 entries only; it does not rewrite sealed Tier-2 raw pages.

3. Provenance Inheritance. If an UPDATE occurs, the new unit inherits the provenance pointers ρ from both the original unit and the new evidence sources, maintaining a complete audit trail.

B.8 Reranking (Optional)

When enabled, we apply a **page-level reranker** to candidate evidence pages. Hits are first grouped by `page_id`; reranking operates on each page’s raw content (or, if missing, concatenated summaries) and selects the top- k *unique pages*. We also include a **protection rule**: pages with high-confidence Mem0 similarity scores are retained to avoid discarding highly relevant memories before reranking.

Discussion: Online Consolidation vs. Epoch-wise Replay Evaluation TierMem is designed to support *online* consolidation: after an escalation, verified findings can be written back immediately to reduce future misses. However, to match the benchmark evaluation protocol and avoid within-run distribution shift, our main experiments report results under an epoch-wise replay setting where Tier-1 is held fixed within each epoch and updates are applied between epochs. We expect that in interactive deployments, moving write-back into the QA loop is straightforward and may further reduce repeated escalations, but careful design is needed to preserve reproducibility and to avoid unintended interactions with evaluation sampling.

B.9 Statistics and Debug Instrumentation

For analysis, we log per-query events including (i) route choice (S vs. R), (ii) hit breakdown by `source_type` (original vs. linked.fact), (iii) linked-fact write-back counts, and (iv) latency/token usage. This instrumentation is used only for evaluation and does not affect the core algorithm.

C Write-Back Replay Table

Table 10 shows the write-back result.

D Router: Data Construction, Training Pipeline, and Hyperparameters

This appendix consolidates all details related to router supervision, SFT distillation, GRPO optimization, and the exact hyperparameters used.

D.1 Task Definition and Notation

The router is a lightweight controller that decides whether the system can answer a user query using Tier-1 summaries alone (ANSWER/S) or must escalate to Tier-2 raw logs

Algorithm 1 TierMem QA (Abstract): Router + Bounded Retrieve–Integrate–Plan Loop

Require: Query Q ; Tier-1 summaries $\mathcal{M}_S$; Tier-2 raw pages $\mathcal{M}_R$
Require: Router π_θ ; generator Gen
Require: Retrieve, Rerank, Integrate, Plan
Require: max iterations $T_{\max}$
Ensure: Answer $\hat{Y}$

```

1:  $Z_0 \leftarrow \text{Retrieve}(\mathcal{M}_S, Q)$ 
2: if  $\pi_\theta(Q, Z_0) = \text{ANSWER}$  then
3:   return Gen( $Q, Z_0$ )
4: end if
5:  $E \leftarrow \text{Integrate}(Q, \text{Linked}(Z_0))$   $\triangleright$  seed: summaries + linked raw pages (compressed)
6:  $q_1 \leftarrow Q; H \leftarrow \emptyset$   $\triangleright$  optional: searched queries / history
7: for  $t = 1$  to  $T_{\max}$  do
8:    $\mathcal{C}_t \leftarrow \text{Retrieve}(\mathcal{M}_S, q_t) \cup \text{Retrieve}(\mathcal{M}_R, q_t)$ 
9:    $X_t \leftarrow \text{TopK}(\text{Rerank}(\mathcal{C}_t, Q))$ 
10:   $E \leftarrow E \cup \text{Integrate}(Q, X_t)$ 
11:   $(d_t, q_{t+1}) \leftarrow \text{Plan}(Q, E, H)$   $\triangleright d_t \in \{\text{STOP}, \text{SEARCH}\}$ 
12:  if  $d_t = \text{STOP}$  then
13:    break
14:  end if
15:   $H \leftarrow H \cup \{q_t\}; q_t \leftarrow q_{t+1}$ 
16: end for
17:  $\hat{Y} \leftarrow \text{Gen}(Q, E)$ 
18:  $\triangleright$  optional: WriteBack( $\mathcal{M}_S, E$ )
19: return  $\hat{Y}$ 
```

Epoch	Overall Acc.	S-Traffic	S-Acc.	Tok _{avg}	Lat.(s)	ADD	UPDATE	DELETE	Tier-1 facts
<i>No-recall write-back (only ADD or SKIP)</i>									
E1	0.854	932	87.9%	3958	5.14	720	–	–	3965
E2	0.847	1125	87.6%	3035	4.16	486	–	–	4451
E3	0.845	1245	87.0%	2419	3.39	317	–	–	4768
<i>Retrieve-and-edit write-back</i>									
E1U	0.854	932	87.9%	3958	5.14	452	209	153	3544
E2U	0.852	1168	87.0%	2719	4.22	312	55	94	3762
E3U	0.853	1224	86.8%	2447	3.85	114	41	33	3843

Table 10: **System evolution under consolidation on LoCoMo** ($n = 1540$). This replay study illustrates qualitative write-back dynamics under epoch-wise consolidation; the main-table LoCoMo router result in Table 2 is **0.854**.

(ESCALATE/R). Given a query Q and an initial evidence state Z_0 consisting of the top- k retrieved Tier-1 summaries, the router predicts:

$$a \sim \pi_\theta(a \mid Q, Z_0), \quad a \in \{S, R\}.$$

The router output is *strictly* a JSON object with an action token (and optionally a teacher thinking field during distillation).

D.2 Data Construction: Summary Sufficiency Labels

Source benchmark slice. We construct router training data from the *Long-Range Understanding* (LRU) competency in MemoryAgentBench Hu et al. (2025), using a summarization-style subset of 2,000 instances. Each instance is materialized into a linked two-tier memory state $(\mathcal{M}_S, \mathcal{M}_R)$ using the write path described in §3.1. For each query Q , Tier-1 dense retrieval returns the initial summary evidence Z_0 (top- k_s summaries), which forms the router input.

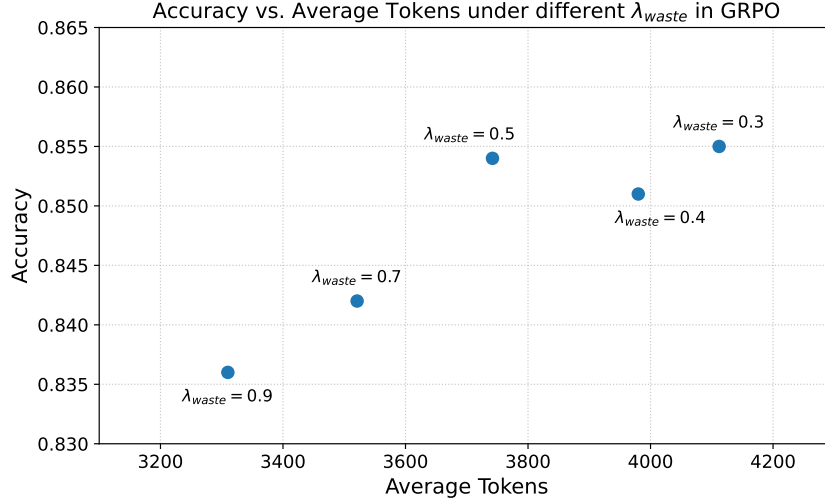


Figure 3: Effect of the GRPO waste-penalty coefficient λ_{waste} on the LoCoMo accuracy–cost trade-off. Moderate penalties reduce unnecessary escalation, but overly large penalties under-route difficult queries and reduce accuracy.

Two-path execution (Summary vs. Research). For each query Q , we execute two fixed policies to determine whether Z_0 is sufficient:

- **Summary-only (S-path):** $\hat{Y}_S \leftarrow \text{Gen}(Q, Z_0)$
- **Research/raw-grounded (R-path):** Run the bounded Deep Search Loop (§3.2) to obtain E_{final} , then $\hat{Y}_R \leftarrow \text{Gen}(Q, E_{\text{final}})$

LLM judge and sufficiency rubric. We obtain binary indicators $c_S, c_R \in \{0, 1\}$ using LLM-based judges with conservative rubrics. Both judges are run with temperature 0 and are constrained to output a fixed JSON schema to reduce variance.

S-path: summary sufficiency judge. Given (Q, y^*, Z_0) , the judge determines whether the retrieved summaries contain *explicitly sufficient* information to answer the question correctly *without guessing*. We use the following prompt (verbatim):

D.3 λ_{waste}

We sweep the waste-penalty coefficient in the GRPO reward to study the accuracy–efficiency trade-off induced by conservative routing. Figure 3 summarizes the resulting operating points. The sweep confirms the expected monotonic trend: larger λ_{waste} discourages escalation and therefore lowers average input-token cost, but overly aggressive penalties begin to suppress useful verification.

Concretely, as λ_{waste} increases from 0.3 to 0.9, average input tokens fall from roughly 4112 to 3310. Accuracy, however, is not monotonic: it peaks around $\lambda_{waste} = 0.5$ at 0.854, remains close at 0.4 (0.851), and then degrades at higher penalties (0.842 at 0.7; 0.836 at 0.9). We therefore use $\lambda_{waste} = 0.5$ in the main results as a balanced point that preserves most of the routing benefit without over-penalizing justified escalation.

D.4 Judge Prompt for Summary Sufficiency

We set $c_S = \mathbb{I}[\text{has_sufficient_info} = \text{true}]$.

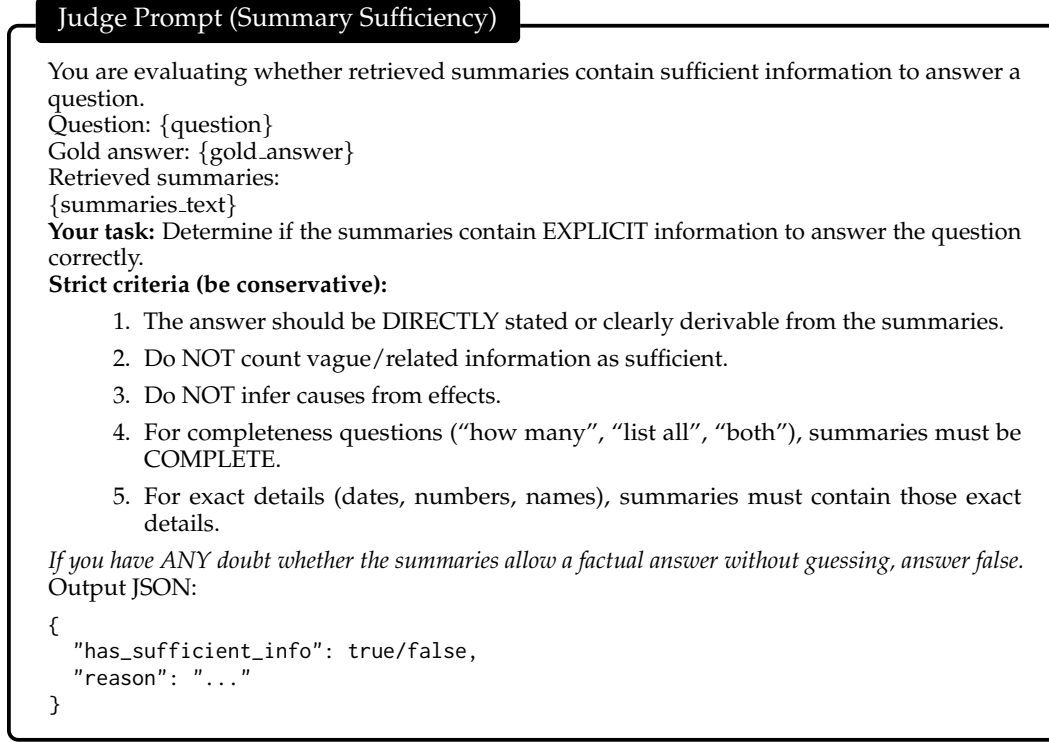


Figure 4: The judge prompt used to evaluate whether the retrieved summaries provide sufficient context to answer the ground truth.

R-path: research/correctness judge. Given $(Q, y^*, \hat{Y}_R)$, the judge determines whether the research-path answer matches the gold answer. We use a separate LLM-as-judge prompt (Figure 12), which outputs a JSON object of the form: $\{ "label": 'CORRECT' / 'WRONG' \}$. We set $c_R = \mathbb{I}[\text{label} = 'CORRECT']$.

Why conservative judging? Our routing label targets *summary sufficiency*. False positives (marking summaries as sufficient when they are not) would train a router that under-escalates and produces ungrounded answers. Therefore, the S-path prompt intentionally biases toward false when there is ambiguity or missing explicit support.

Routing labels (semantic cache hit/miss). We define the router supervision label as:

$$y = \begin{cases} S & \text{if } c_S = 1, \\ R & \text{if } c_S = 0 \wedge c_R = 1. \end{cases}$$

We drop instances where both policies fail ($c_S=0 \wedge c_R=0$), since they provide no signal on whether escalation can recover missing evidence.

D.5 Training Workflow Overview

We train the router using a two-stage pipeline:

1. **Stage 1 (SFT / Distillation):** Initialize the router to follow the routing protocol and learn strong sufficiency heuristics from a high-capacity teacher (GPT-5) using a strict JSON format.
2. **Stage 2 (GRPO):** Starting from the SFT checkpoint, refine the decision boundary under an explicit accuracy–cost trade-off, discouraging unnecessary escalation while preserving faithfulness.

This separation is intentional: SFT primarily regularizes behavior and format; GRPO optimizes the accuracy–efficiency trade-off.

D.6 Stage 1: SFT via GPT-5 Thinking+Action Distillation

Teacher prompt (GPT-5 thinking trace + binary action). We distill a teacher-generated thinking trace and the final routing decision action using the following prompt (verbatim):

D.7 Router Thinking Prompt

Router Thinking Prompt

You are a router for a memory-augmented QA system. Decide whether the retrieved summaries are sufficient to answer the question.

Available Actions

1. **“S” — Answer using current summaries only**
 Use when: The summaries contain a span that directly states the answer (or a direct paraphrase) WITHOUT needing to assume missing facts.
Rules of thumb:
 - If you can point to a specific snippet that can be copied into the answer, choose S.
2. **“R” — Deep research mode**
 Use when: Summaries are too ambiguous, contradictory, or unrelated; OR the question requires a comprehensive list and summaries are clearly incomplete.
 R is more reliable than S.

Input
 QUESTION: {question}
 SUMMARIES:
 {summaries.block}

Decision checklist

- What exact answer format is needed? (single value / list / date / identity)
- Do summaries explicitly contain it?
- Any ambiguity or need to infer?
- Any completeness requirement?

Output (JSON only)

```
{
  "thinking": "<your reasoning>",
  "action": "S/R"
}
```

Figure 5: The updated router prompt incorporating explicit chain-of-thought reasoning to improve decision reliability.

SFT targets and formatting. For each example, the student router is trained to reproduce the teacher’s JSON response. We treat the entire JSON (including thinking and action) as the supervised target, using standard next-token cross-entropy. At inference time, the system uses only the action field.

Teacher-label consistency filter. To ensure alignment with oracle labels derived in §D.2, we keep only SFT examples where the teacher action matches the oracle routing label y . Examples with mismatched teacher actions are discarded.

Why include thinking in SFT. The teacher thinking field provides structured intermediate reasoning about evidence sufficiency (e.g., completeness, ambiguity, exactness), which

improves stability and reduces instruction-following failures (e.g., verbose or malformed outputs) in a small router.

D.8 Stage 2: GRPO Alignment for Accuracy–Cost Trade-off

Starting from the SFT checkpoint, we optimize the router with GRPO using a trajectory-level reward:

$$R(\tau) = R_{\text{acc}}(\tau) - \lambda_{\text{cost}} R_{\text{cost}}(\tau) - \lambda_{\text{waste}} R_{\text{waste}}(\tau), \quad (1)$$

where:

- R_{acc} encourages correct routing (choose S when summaries suffice; choose R when escalation is necessary),
- R_{cost} penalizes choosing R to reflect token/latency overhead,
- R_{waste} penalizes *unnecessary escalation* (choosing R when $y = S$).

Concrete reward implementation. In our implementation, rewards are computed from (c_S, c_R) and the router decision a .

GRPO dataset construction. We build GRPO prompts using the same router input (Q, Z_0) as in SFT. For each prompt, we sample multiple router completions per prompt (generations per prompt is reported in §D.10) to compute group-relative advantages.

D.9 GRPO Dataset Preparation and Filtering

Starting from the offline QA logs (which include both summary-path and research-path outcomes), we apply the following filters:

1. Remove instances where **both** paths fail ($c_S = 0 \wedge c_R = 0$).
2. Remove instances where the **R-path fails** ($c_R = 0$), since escalation does not recover evidence.
3. Optionally re-judge S-path sufficiency using a fixed judge configuration to reduce label noise; keep only instances consistent under re-judging.

Optional augmentation. We optionally augment a subset of questions by paraphrasing while preserving intent and specificity (2–3 paraphrases per question). Paraphrases inherit the same memory state and labels.

Balancing. To emphasize hard decisions, we oversample cases where escalation is necessary ($c_S = 0 \wedge c_R = 1$) during GRPO minibatch formation.

D.10 Hyperparameters

D.10.1 SFT Hyperparameters

Parameter	Value
Train type	LoRA
Precision	bfloat16
Epochs	5
Learning rate	1×10^{-4}
Max sequence length	4096
Per-device train batch size	8
Per-device eval batch size	4
Gradient accumulation steps	1
LoRA rank / alpha	64 / 128
Target modules	all-linear
Warmup ratio	0.1
Eval / Save steps	10 / 10
Save total limit	5
Logging steps	1
Dataloader workers	4
Loss scaling	ignore_empty_think
Kernel optimization	use_liger_kernel=true
DeepSpeed	ZeRO-2

Table 11: SFT (LoRA) training configuration for the router.

D.10.2 GRPO Hyperparameters

Parameter	Value
RLHF type	GRPO
Initialization	SFT adapter checkpoint
Precision	bfloat16
Epochs	6
Context length	2048
Learning rate	5e-6
Max completion length	256
Per-device train batch size	24
Per-device eval batch size	24
Gradient accumulation steps	1
Warmup ratio	0.05
DeepSpeed	ZeRO-2
Target modules	all-linear
LoRA rank / alpha	64 / 128
Dataloader workers	4
Logging / Save steps	5 / 50
Completions logging	log_completions=true
Sampling temperature	1.2
Generations per prompt	8
Rollout forward batch size	8
KL / regularization parameter	$\beta = 0.02$
Reporting	Weights & Biases

Table 12: GRPO training configuration for router policy optimization.

D.10.3 Reward Coefficients

Name	Value
Correct reward (CORRECT_REWARD)	1.0
Wrong penalty (WRONG_PENALTY)	-1.5
Cost of S (COST_S)	0.0
Cost of R (COST_R)	0.2
Waste penalty for unnecessary R (WASTE_R)	0.5
Format error penalty (FORMAT_ERROR_PENALTY)	-1.0

Table 13: Reward coefficients used by the router GRPO objective.

E Detailed Analysis of Provenance Pointer Ablation

This appendix provides a detailed analysis of the provenance pointer ablation study introduced in Section 4.3.1. We analyze route consistency, accuracy under identical routing decisions, and token-level cost, in order to isolate the effect of provenance-aware warm-start from routing behavior or additional computation.

E.1 Route Consistency

Across 1,540 shared queries, Linked and No-Linked make identical routing decisions on 1,306 queries (84.8%), and differ on 234 queries (15.2%). This indicates that enabling provenance pointers induces only modest changes in routing behavior. Therefore, improvements observed in the Linked setting cannot be explained solely by more aggressive escalation, but instead suggest higher-quality evidence retrieval during escalation.

E.2 Accuracy under Identical Routes

To disentangle retrieval quality from routing effects, we analyze queries where Linked and No-Linked select the same route.

Both Escalate (R). When both systems escalate to raw logs, Linked answers 31 queries correctly that No-Linked answers incorrectly, whereas the reverse occurs in 25 cases. This yields a net gain of +6 queries for Linked under identical escalation decisions.

Both Answer (S). When both systems answer directly from summaries, Linked answers 16 queries correctly that No-Linked answers incorrectly, compared to 10 in the opposite direction. The small difference confirms that provenance pointers primarily affect raw evidence grounding rather than summary-only answering.

These controlled comparisons demonstrate that provenance pointers improve the *quality* of retrieved evidence, independent of routing frequency.

E.3 Cost Analysis by Route

Table 14 reports average token usage by route. On summary-only queries, Linked and No-Linked incur nearly identical cost. On escalated queries, Linked incurs slightly higher token usage, reflecting deeper or more precise raw evidence grounding.

Importantly, the average research depth is nearly identical between Linked (1.38) and No-Linked (1.36), with the same maximum depth (3), indicating that gains do not stem from additional retrieval iterations.

Method	Route	Tok _{in}	Tok _{out}
Linked	S	847.5	97.8
Linked	R	8875.1	724.0
No-Linked	S	847.2	96.8
No-Linked	R	8705.0	709.9

Table 14: Average QA token cost by route for provenance pointer ablation.

User Query	Tier-1 Summary (Compressed)	Router Decision & Reasoning
<i>Case 1: Mitigating Entity Loss (The “Hero” Case)</i>		
Q: “What book recommendations has Joanna given to Nate?”	Context: “...Joanna recommended Nate to find a fantasy book series ... Joanna approved the specific book series Nate showed her...” XDefect : The specific entity (book title) is lost during summarization.	Decision : ESCALATE (Raw Search) Router Thought : “Summaries mention Joanna recommended a series, but do not explicitly state the book titles . They are contextually related but lack the specific answer.” Outcome : Retrieved raw log containing “ <i>Little Women</i> ”, avoiding a vague or hallucinated answer.
<i>Case 2: Semantic Sufficiency (The “Saver” Case)</i>		
Q: “What important values does John want to teach his kids through adopting a rescue dog?”	Context: “...John said adopting a rescue dog teaches John’s kids responsibility and compassion ...” ✓Sufficient : Key semantic concepts are present.	Decision : ANSWER (Summary Only) Router Thought : “Summaries do not literally state ‘values: X’, but they contain the needed semantic content. Since the answer is present , use Summary.” Outcome : Answered correctly while skipping several thousand tokens of raw retrieval.

Table 15: Qualitative case studies illustrating how TierMem aligns retrieval granularity with query requirements.

F Case Study: Qualitative Cases

We present two representative cases illustrating how TierMem aligns retrieval granularity with query requirements in Table F. Case 1 requires escalation due to entity loss in summarization, and Case 2 can be answered cheaply from summaries because key semantic evidence is preserved. A broader analysis of router decision errors is provided in Section G.

Mitigating compression loss. When the user query requires a specific entity that is missing from Tier-1 summaries, TierMem escalates and grounds the answer in Tier-2 raw logs, avoiding vague or hallucinated responses.

Ensuring semantic efficiency. When Tier-1 summaries contain semantically sufficient evidence (even if not phrased in the same surface form as the query), the router answers directly, avoiding unnecessary raw retrieval while preserving correctness.

G Router Error Analysis and Failure Modes

We analyze routing errors relative to the raw-grounded reference and cluster them into coherent categories (Table 16). We distinguish two asymmetric error types: (i) **S→R errors** (false cache hits), where the router answers from Tier-1 but raw grounding is required; and (ii) **R→S errors** (false cache misses), where the router escalates unnecessarily despite sufficient summary evidence.

False cache hits are rarer but more harmful. S→R errors are relatively rare (72 cases) but directly impact correctness. They stem mainly from *information completeness* failures (missing

Error Category	Count
S→R Errors (Summary insufficient, escalation needed)	
Information completeness error	30
Disambiguation failure	22
Precision requirement ignored	16
Context and focus confusion	1
Other	3
Total S→R	72
R→S Errors (Unnecessary escalation)	
False information gap detection	204
Neglect of indirect or inferential evidence	52
Temporal or logical inference failure	36
Total R→S	292

Table 16: **Router error analysis.** We categorize routing errors into false cache hits (S→R) and false cache misses (R→S). False cache hits are relatively rare but stem mainly from summary incompleteness and unmet precision requirements. False cache misses dominate and primarily reflect conservative over-escalation, incurring additional cost without affecting correctness.

a required slot such as an entity name) and *precision requirement* failures (ignoring that the question demands exactness or attribution). These cases align with known brittleness of lossy compression: a summary can be topically relevant yet underspecified for exact answering.

False cache misses dominate and reflect conservative bias. R→S errors are much more frequent (292 cases). The primary cause is *false information gap detection*: the router assumes summaries are insufficient even when they contain enough evidence. These errors increase cost but typically do not degrade correctness, reflecting a conservative policy that prioritizes faithfulness. This suggests a clear path for future improvement: better calibration and sufficiency detection that leverages indirect but adequate evidence while still avoiding false cache hits.

H Prompt Templates

Fact Extraction Prompt (JSON Schema)

You are the **Universal Memory Encoder**. Your goal is to convert raw input stream into high-fidelity, self-contained knowledge records (Long-term Memory).

INPUT FORMATTING NOTICE:

The input will be provided in the next user message, prefixed by 'Input:' and often formatted as dialogue lines.

It may follow a pattern like: '[Timestamp] Speaker_Name: Content'.

- If this pattern is present, you **MUST** use the 'Timestamp' for temporal grounding and 'Speaker_Name' for entity resolution.
- If this pattern is absent (e.g., raw document text), treat the input as a factual source and extract knowledge propositions.

CORE OBJECTIVES:

1. ENTITY & CONTEXT RESOLUTION:

- **No ambiguous references:** Every extracted fact must explicitly name the involved people, organizations, objects, places, and concepts.
- **No "User" ambiguity:** If the input says '[...] Melanie: I like art', the fact **MUST** be 'Melanie likes art', NOT 'The user likes art'.
- **De-contextualization:** Each extracted fact must be **standalone**.

2. TEMPORAL GROUNDING:

Use the provided timestamps in the input as the source of truth. Convert relative time ('tomorrow', 'next week') into absolute context.

3. PRONOUN & DEIXIS ELIMINATION:

Extracted facts must **not contain any pronouns**. If a pronoun appears, resolve it to the specific entity.

4. SCENE TAGGING (EMBEDDED):

Identify the immediate scene/activity (2-5 words). Append this scene to the end of the fact string inside brackets.

OUTPUT INSTRUCTIONS (MUST FOLLOW):

Return a JSON object with **EXACTLY** this schema:

```
{
  "facts": [
    "<atomic fact sentence 1> [Scene: ...]",
    "<atomic fact sentence 2> [Scene: ...]", ...
  ]
}
```

Rules: Output JSON only (no extra text, no markdown code fences).

Figure 6: The prompt used for extracting atomic facts from conversation streams.

Research Integration Prompt

You are a research assistant extracting facts from conversation evidence to answer a question.

QUESTION: {question}

RETRIEVED EVIDENCE: {evidence}

YOUR TASK:

Read through ALL pages in the evidence and extract facts that help answer the question.

EXTRACTION STRATEGY:

Identify what information is needed and extract:

1. **Direct answers:** Facts that directly answer the question
2. **Component facts:** Facts about entities/topics in the question that can be combined
3. **Temporal facts:** When events happened
4. **Confirmation facts:** Look for acceptance/approval messages

OUTPUT FORMAT (JSON):

```
{
  "linked_facts": [
    {
      "fact": "<extracted fact - use specific dates/names>",
      "evidence_quote": "<EXACT quote from the evidence>"
    }
  ],
  "coverage_assessment": "<what aspects are covered vs missing>"
}
```

Figure 7: Prompt for integrating retrieved facts to answer a query.

Research Plan Prompt

You are a research planner. Based on the current facts and the question, decide if more information is needed.

QUESTION: {question}

CURRENT FACTS: {current_facts}

COVERAGE ASSESSMENT: {coverage_assessment}

RESEARCH HISTORY: {research_history}

ALREADY SEARCHED: {searched_queries}

YOUR TASK:

Decide whether the current facts are SUFFICIENT to answer the question, or if more search is needed.

OUTPUT FORMAT (JSON):

```
{
  "decision": "DONE" or "SEARCH",
  "reasoning": "<brief explanation>",
  "search_commands": [
    { "type": "MEMO_SEARCH", "query": "<semantic search query>" },
    { "type": "KEYWORD_SEARCH", "keywords": ["k1", "k2"] }
  ]
}
```

DECISION CRITERIA:

- **DONE:** The facts contain enough information to provide a reasonable answer.
- **SEARCH:** Key information is missing and more search might help.

Figure 8: Prompt for planning the next research step or concluding the search.

Router Prompt (Summarization vs. Research Decision)

You are an expert router for a memory-augmented QA system. Analyze the retrieved summaries and decide the best action to answer the question.

Available actions:

1. "S" - Answer using current summaries only. Use when summaries contain the EXPLICIT answer.
2. "R" - Deep research mode (slow path). Use when summaries are ambiguous, only contextually related, or miss the answer entirely.

Question: {q}

Retrieved Summaries: {summaries.block}

Output format (JSON only):

- If answering with summaries: {"action": "S"}
- If deep research needed: {"action": "R"}

Figure 9: Routing prompt to decide between immediate summarization or deep research.

Answer Generation Prompts (Template Functions)

```
def _make_locomotive_summary_prompt(summary: str, question: str) -> str:
    return f"""
Based on the summary below, write an answer in the form of a short phrase...
Answer with exact words from the context whenever possible.
For date/time, strictly follow the format "15 July 2023"...

QUESTION: {question}
SUMMARY: {summary}
Short answer:
"""

def _make_longmemory_prompt(summary: str, question: str) -> str:
    return f"""
You are an intelligent memory assistant tasked with retrieving accurate
information from episodic memories.

# INSTRUCTIONS:
Synthesize information from different memories to answer the user's question.
It is CRITICAL that you move beyond simple fact extraction and perform
logical inference. Answer the question in a short phrase.

QUESTION: {question}
SUMMARY: {summary}
Short answer:
"""
```

Figure 10: Python functions used to generate the final answer prompts based on context.

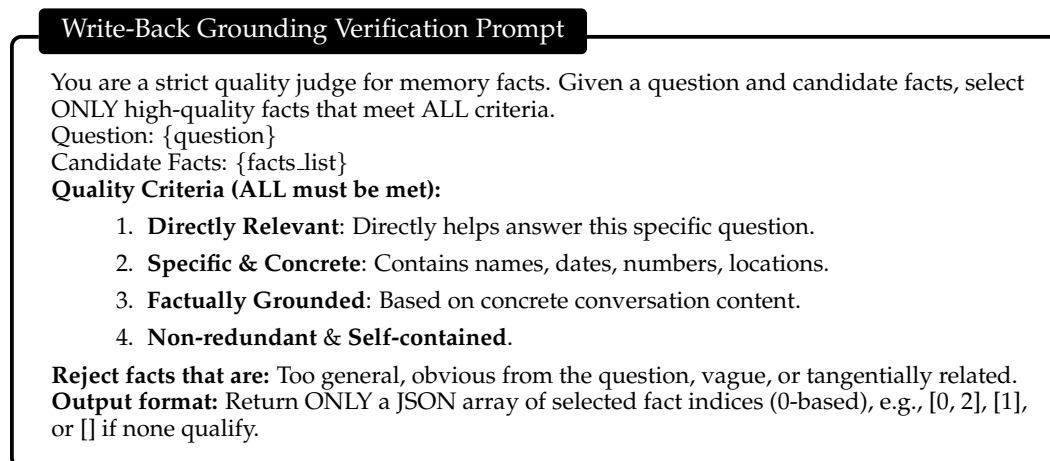


Figure 11: Prompt for verifying the quality and relevance of extracted facts.

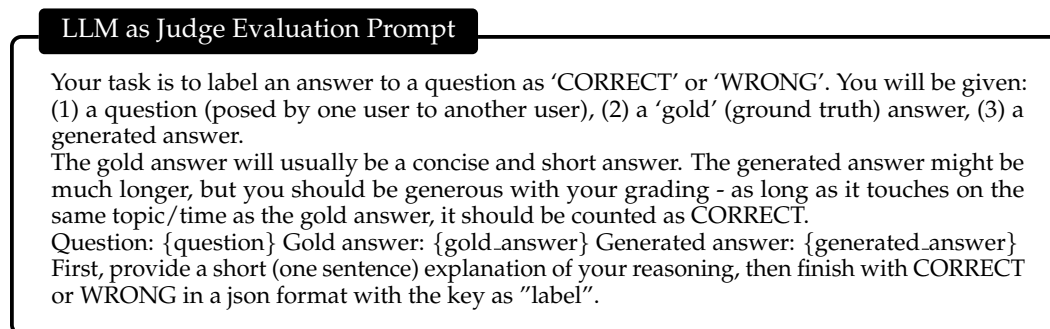


Figure 12: The LLM-as-a-Judge prompt used for automated evaluation.